Approximating Nash Equilibria for Black-Box Games: A Bayesian Optimization Approach*

Abdullah Al-Dujaili, Erik Hemberg, and Una-May O'Reilly

CSAIL, MIT, USA
aldujail@mit.edu, {hembergerik,unamay}@csail.mit.edu

Abstract. Game theory has emerged as a powerful framework for modeling a large range of multi-agent scenarios. Many algorithmic solutions require discrete, finite games with payoffs that have a closed-form specification. In contrast, many real-world applications require modeling with continuous action spaces and black-box utility functions where payoff information is available only in the form of empirical (often expensive and/or noisy) observations of strategy profiles. To the best of our knowledge, few tools exist for solving the class of expensive, black-box continuous games. In this paper, we develop a method to find equilibria for such games in a sequential decision-making framework using Bayesian Optimization. The proposed approach is validated on a collection of synthetic game problems with varying degree of noise and action space dimensions. The results indicate that it is capable of improving the game-theoretic regret in noisy and high dimensions to a greater extent than hierarchical or discretized methods.

Keywords: Game Theory · Pure Strategy · Empirical Games · Black-Box Optimization · Gaussian Processes · Nash Equilibrium.

1 Introduction

Game-theoretic solution concepts play an important role in understanding agents interactions in various areas including ecnonomics [10], politics [21], and cybersecurity [17]. The problem of approximating game equilibria and strategic stability has received a lot of attention in the literature. Many of the present solution algorithms and approximation tools are tailored to a restricted class of games. Complex games are then either stylized into a version covered by available solvers (e.g., coarse discretization for infinite games [15]) or estimated through simulation and sampling (empirical game-theoretic analysis [25,31]). Solving finite approximations to an infinite game can be instructive, but also produce misleading results [26]. On the other hand, analyzing games, through simulation and sampling, poses a *search problem* with the goal of identifying equilibrium profiles given a finite number of payoff (utility function) evaluations. In this

* This material is based upon work supported by the MIT-IBM Watson AI Lab and the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR0011-16-C-0101.

paper, our interest is the search problem when such evaluations are expensive and/or noisy.

We consider one-shot non-cooperative normal-form games, where players make decisions about their actions simultaneously and receive payoffs, upon which the game ends. Our goal is a general-purpose Nash equilibrium approximation technique for such games that are expensive, black-box, and continuous. We report the following contributions: 1) To the best of our knowledge, this is the first work that approximates equilbria in an expensive, black-box context on the full continuous action space, rather than a discretized representation of the same. 2) While majority of equilibria search methods for black-box games cast the problem as the bi-level optimization relying on a best-response search subroutine, we approximate best-responses based on the learned payoff functions. Subsequently, our method solves a flat optimization problem, rather than a hierarchical one. This saves significant computational expense. 3) We validate the effectiveness of our proposition and compare its performance in terms of regret against recent algorithms in the literature on a collection of synthetic games. With about 25 function evaluations or more, the proposed algorithm is capable of minimizing the regret to an extent which existing tailored methods are not able to reach 4) Finally, we provide an implementation for public use.

2 Background

This section introduces the main terminology—which we adopt from [13,28,23]—used in the rest of the paper, followed by a summary of related work.

2.1 Formal Background

We start with formalizing the notion of strategic interactions between a set of rational players as follows.

Definition 1. *(Normal Form Game [29])* $(I, \{\mathcal{X}_i\}, \{u_i(\boldsymbol{x})\})$ *is a normal form game, with* I *the set of players where* $|I|= p$*,* $\mathcal{X}_i$ *the set of strategies (pure or mixed, depending on context) available to player* i*,* $\mathcal{X} = \prod_{i=1}^{p} \mathcal{X}_i \subseteq \mathbb{R}^{n_{\mathcal{X}}}$ *the joint strategy set, and* $u_i : \mathcal{X} \to \mathbb{R}$ *the utility function for player* i *mapping the joint strategy* $\boldsymbol{x}$ *to the real-valued payoff received by player* i *when* $\boldsymbol{x}$ *is played.*

With *pure strategies*, players choose their *actions* deterministically in the game. Thus, the joint strategy $\boldsymbol{x} \in \mathcal{X}$ represents the joint actions of the set of players I, and the term *action* is interchangeable with *strategy*. On the other hand, *mixed strategies* are probability distributions over pure strategies. For brevity, we sometimes omit the word *pure* when referring to a pure (joint) strategy in the rest of the paper. Further, the term *joint strategy* is interchangeable with *profile*.

We shall use the convention $\boldsymbol{x} = (\boldsymbol{x}_i, \boldsymbol{x}_{-i})$ when the role of player i's action $\boldsymbol{x}_i$ needs to be emphasized. Accordingly, for player i, we have $u_i(\boldsymbol{x}) = u_i(\boldsymbol{x}_i, \boldsymbol{x}_{-i})$. Likewise, $\mathcal{X} = \mathcal{X}_i \times \mathcal{X}_{-i}$. Holding other players' strategies $\boldsymbol{x}_{-i}$ constant, player

i can best respond by unilaterally deviating from her current strategy $\boldsymbol{x}_i$ such that her payoff is maximized, as described formally below.

Definition 2. *(Best Response [13]) For some joint strategy $\boldsymbol{x} \in \mathcal{X}$, the player i's best-response correspondence is*

$$\mathcal{B}_i(\boldsymbol{x}) = \arg\max_{\boldsymbol{x}'_i \in \mathcal{X}_i} u_i(\boldsymbol{x}'_i, \boldsymbol{x}_{-i}) \ . \tag{1}$$

The joint best-response correspondence is then given by

$$\mathcal{B}(\boldsymbol{x}) = \prod_{i \in I} \mathcal{B}_i(\boldsymbol{x}) \ . \tag{2}$$

The notion of best response is related to another important measure of normal-form games, viz. the *game-theoretic regret* (or simply *regret*). Denoted by $\epsilon(\boldsymbol{x})$, the regret of $\boldsymbol{x}$ is the most any player i can gain by deviating from $\boldsymbol{x}_i$ to any strategy in $\mathcal{X}_i$. Mathematically, we have

$$\epsilon(\boldsymbol{x}) = \max_{i \in I} u_i(\mathcal{B}_i(\boldsymbol{x}), \boldsymbol{x}_{-i}) - u_i(\boldsymbol{x}) \ . \tag{3}$$

Iterative application of best-response correspondence results in the *best-response dynamic*, for which a pure-strategy *Nash equilibrium* (NE) $\boldsymbol{x}^*$ is a fixed point. That is, $\boldsymbol{x}^* = \mathcal{B}(\boldsymbol{x}^*)$. Nash equilibrium is a game-theoretic solution concept for non-cooperative static games with complete information and no leadership nor followers features [10]. It formalizes the notion of strategic stability in the sense that every player is playing optimally given other players' choices, as defined next.

Definition 3. *A Nash equilibrium $\boldsymbol{x}^* \in \mathcal{X}$ is the joint strategy such that*

$$\forall i \in I \ , \ \ \boldsymbol{x}^*_i = \mathcal{B}_i(\boldsymbol{x}^*) \ , \tag{4}$$

or equivalently,

$$\epsilon(\boldsymbol{x}^*) = 0 \ . \tag{5}$$

A pure NE does not always exist. Moreover, a game can sometimes be too large to compute a NE exactly, or only payoff estimates are available. For these cases, $\boldsymbol{\varepsilon}$-NE is an appropriate solution concept. A profile $\boldsymbol{x} \in \mathcal{X}$ is an $\boldsymbol{\varepsilon}$-NE if and only if $\epsilon(\boldsymbol{x}) \leq \boldsymbol{\varepsilon}$, where $\boldsymbol{\varepsilon} \geq 0$. It follows that every profile $\boldsymbol{x}$ is an $\epsilon(\boldsymbol{x})$-NE, and the pure NE $\boldsymbol{x}^*$ is a 0-NE.

In this paper, we are interested in approximating *pure NEs for continuous black-box games*: i) By *continuous*, we mean that $\{\mathcal{X}_i \subseteq \mathbb{R}^{n_{\mathcal{X}_i}}\}$. That is, player i's actions are real-valued vectors of size $n_{\mathcal{X}_i}$. ii) By *black-box*, we mean that there is no closed-form expression of the utility functions $\{u_i\}$, or their gradients are neither symbolically nor numerically available. Instead, one can query an oracle $\boldsymbol{o}$ (e.g., a simulation), which produces a possibly noisy version of $\{u_i\}$ at specific profile $\boldsymbol{x}$. Each oracle call—also referred to as a function evaluation (`FE`)—is often expensive in terms of computational resources (e.g., CPU time). Denoted by $(I, \{\mathcal{X}_i\}, \boldsymbol{o})$, such games are also referred to as *empirical* or *simulation-based* games [29,28]. Mathematically, we have $\boldsymbol{o}(\boldsymbol{x}) = (o_1(\boldsymbol{x}), \ldots, o_p(\boldsymbol{x}))$ and

$\mathbb{E}[\boldsymbol{o}(\boldsymbol{x})] = \boldsymbol{u}(\boldsymbol{x})$, where $\boldsymbol{u} = (u_1(\boldsymbol{x}), \ldots, u_p(\boldsymbol{x}))$. Further, we use the notation $\boldsymbol{o}^{(t)}$ and $o_i^{(t)}$ to denote $\boldsymbol{o}(\boldsymbol{x}^{(t)})$ and $o_i(\boldsymbol{x}^{(t)})$, respectively. Similar notation will be used for other quantities. In the next section, we summarize literature related to approximating equilibria (if one exists) for expensive, black-box, continuous game-theoretic models.

2.2 Related Work

Continuous Games. For many years, AI researchers have worked on techniques for approximating equilibria in *finite* games (finite action spaces $\mathcal{X}$). For instance, the Gambit [19] software package offers a collection of established algorithms to find NEs for finite games. Similarly, theoretical and algorithmic works on *continuous* and *infinite* games are an area of active research. Debreu [6] showed that pure-strategy equilibria exist for infinite games of complete information with compact, convex action spaces (subsets of a Euclidean space $\mathbb{R}^{n_{\mathcal{X}}}$) and payoffs that are continuous and quasiconcave in the actions. Fixed-points methods for computing NEs, using standard nonlinear programming routines, were extensively studied [3,16,27,14]. Some propositions employed coarse discretization amenable to finite-game solvers [15], while others were tailored to restricted classes of games. For instance, an algorithm was presented in [26] for two-player games with a class of piecewise linear utility functions.

Black-Box Continuous Games. Analytical tools of game theory have been applied to games that are constructed from empirical observations of strategic play [31]. The source of these observations is an oracle $\boldsymbol{o}$ representing agent-based simulation or real-world data. The main challenge is to develop algorithms capable of approximating NEs without any information about the payoffs $\{u_i\}$, except for a set of t profile-wise observations $\mathcal{D}^{1:t} = \{(\boldsymbol{x}^{(1)}, \boldsymbol{o}^{(1)}), \ldots, (\boldsymbol{x}^{(t)}, \boldsymbol{o}^{(t)})\}$. Vorobeychik et al. [30] studied approximating utility functions $\{u_i\}$ given a random sample of pure profiles using supervised learning (regression) methods. Success was measured by how close the players' predicted behavior to that associated with the true utility functions. The $\boldsymbol{o}$-query complexity (i.e., the number of function evaluations #`FEs`) of learning equilibria for various classes of games was theoretically addressed in [8]. For oracles with noisy outcomes, [29] presented a convergent (in probability) algorithm, which approximates Nash equilibria by minimizing the game-theoretic regret (Eq. 3), based on a hierarchical application of Simulated Annealing (SA) and Monte Carlo methods (to estimate expected payoffs). It was shown empirically that a hybrid method of SA and the Harmony Search algorithm [9] converges to an approximate equilibrium point faster than the plain SA at the expense of a slightly lower approximation accuracy [1].

Expensive Black-Box Continuous Games. To the best of our knowledge, very little literature addresses game-theoretic models that are expensive to evaluate. The aforementioned hierarchical SA becomes impractical when each oracle call (or simulation) takes nearly an hour [32]. At the time of writing this paper, a new version of an article [23] was released on arXiv proposing a novel approach

for approximating equilibria in a continuous, black-box, expensive context using Bayesian Optimization (BO) [20], which is an established technique to optimize black-box problems. Their proposed approach is realized by fitting Gaussian Processes (GPs) over a coarse discretization of the action space $\mathcal{X}$. This is the most relevant work to our proposition and shares the same goal. The main differences are i) *Representation of action spaces* $\mathcal{X}$: we seek to identify and learn the full game from the limited oracle calls in a way similar to [30], while [23] assumes that $\mathcal{X}$ is either originally discrete, or a representative discretization is available. ii) *BO acquisition function*: we employ BO in minimizing an *approximate* of the game-theoretic regret (Eq. 3) in a way similar to [29], whereas in [23], BO is used to maximize the joint probability of achieving equilibrium or minimizing an uncertainty measure related to equilibrium.

3 Methods

In this section, we describe our proposition, which we refer to as `BN`: Bayesian optimization to approximate Nash equilibria given a finite number of oracle $\boldsymbol{o}$ queries. The approach we take is similar to [29]: to minimize the regret (Eq. 3). As shown in Figure 1, regret-minimization approaches usually employ best-response $\{\mathcal{B}_i(\boldsymbol{x})\}$ approximation as a subroutine to estimate the regret $\epsilon(\boldsymbol{x})$ at a given profile $\boldsymbol{x} \in \mathcal{X}$. This entails solving a bi-level optimization problem, which can require a prohibitive number of function (oracle) evaluations. This poses a challenge in expensive settings. Instead of using best-response approximation as a

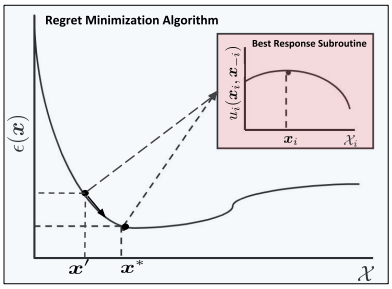


Fig. 1. A diagramatic view of finding Nash equilibrium based on regret minimization. Adapted from [29].

subroutine [29], we make use of GPs to jointly learn the payoffs and approximate the regret based on $\mathcal{D}^{1:t}$: the set of empirical observations of strategic plays obtained after t function evaluations (`FEs`). At the same time, we employ the BO framework to search for a NE $\boldsymbol{x}^*$ by sampling the profile space $\mathcal{X}$ sequentially. The sequential sampling is guided by optimizing the approximated regret $\hat{\epsilon}(\cdot|\mathcal{D}^{1:t})$, which represents the so-called *acquisition function* in BO literature.

Optimizing $\hat{\epsilon}$ can be carried out by off-the-shelf global optimization algorithms, as it does not make any call to the oracle $\boldsymbol{o}$ and hence it is relatively inexpensive to evaluate. Mathematically, at step $t \geq t_o$,

$$\boldsymbol{x}^{(t+1)} \in \arg\min_{\boldsymbol{x}\in\mathcal{X}} \hat{\epsilon}(\boldsymbol{x}|\mathcal{D}^{1:t}).$$

The profile $\boldsymbol{x}^{(t+1)}$, which is a potential NE, represents the next strategic play to be evaluated by invoking the oracle $\boldsymbol{o}$. At step $t+1$, the GP models are fitted with

$$\mathcal{D}^{1:t+1} = \mathcal{D}^{1:t} \cup \{(\boldsymbol{x}^{(t+1)}, \boldsymbol{o}(\boldsymbol{x}^{(t+1)}))\} \ ,$$

and the procedure continues iteratively. After T `FEs`, the algorithm returns the profile $\boldsymbol{x}(T)$ of the lowest obtained regret from the constructed sequence $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(T)}\}$. Note that, $\boldsymbol{o}(\boldsymbol{x})$ corresponds to a single (possibly noisy) sample of the payoffs value at profile $\boldsymbol{x}$, in contrast to the Monte Carlo estimate (averaging several samples) employed in the hierarchical SA approach of [29]. We consider models with additive noise corruption, that is

$$o_i(\boldsymbol{x}) = u_i(\boldsymbol{x}) + \zeta_i \ , \ \forall i \in I, \tag{6}$$

where $\zeta_i \sim \mathcal{N}(0, v_i)$ with $v_i \geq 0$.

A GP is a distribution over functions specified by its mean $\mu(\cdot)$ and covariance (or kernel) $k(\cdot,\cdot)$ functions. Given the profiles $(\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(t)})$, denoted by $\boldsymbol{x}^{1:t}$, and the corresponding observed player i's payoffs $(o_i^{(1)}, \ldots, o_i^{(t)})$, denoted by $o_i^{1:t}$, we have

$$o_i^{1:t} \sim \mathcal{N}\left(\mu_i^{1:t}, K_i\right) \ , \ \forall i \in I \ , \tag{7}$$

where $\mu_i^{1:t} = (\mu_i^{(1)}, \ldots, \mu_i^{(t)})$ and K_i is the covariance matrix, with its entries $K_i^{(p,q)} = k_i(\boldsymbol{x}^{(p)}, \boldsymbol{x}^{(q)})$. Common choices of $\{k_i(\cdot,\cdot)\}$ in BO literature are the squared exponential and Matérn kernels. For more details on GPs, we refer the reader to [24]. Our choice of GPs was primarily due to their analytical tractability, which enable us to compute exactly the posterior predictive mean $\mu_i(\cdot|\mathcal{D}^{1:t})$ and variance $\sigma^2(\cdot|\mathcal{D}^{1:t})$ for any profile $\boldsymbol{x} \in \mathcal{X}$, as well as to approximate the regret $\epsilon(\boldsymbol{x})$ (Eq. 3) by $\hat{\epsilon}(\cdot|\mathcal{D}^{1:t})$. Recall that computing the regret requires the values $u_i(\boldsymbol{x}_i, \boldsymbol{x}_{-i})$ and $u_i(\mathcal{B}(\boldsymbol{x}), \boldsymbol{x}_{-i})$ for all $i \in I$. In the literature [29,28], the former quantity has been estimated by invoking the oracle multiple times for $\boldsymbol{x}$ and averaging the obtained observations, while the latter quantity is estimated by a stochastic search method (e.g., SA), which also invokes the oracle multiple times at each point of its search trajectory. Here, we approximate $u_i(\boldsymbol{x}_i, \boldsymbol{x}_{-i})$ and $u_i(\mathcal{B}(\boldsymbol{x}), \boldsymbol{x}_{-i})$ with GPs.

Approximating $u_i(\boldsymbol{x}_i, \boldsymbol{x}_{-i})$. Given $\mathcal{D}^{1:t}$ and assuming that the prior mean function $\mu_i(\cdot) = 0$, the posterior predictive distribution of $o_i(\boldsymbol{x}^{(t+1)})$ is of the form [24]

$$o_i^{(t+1)}|\mathcal{D}^{1:t} \sim \mathcal{N}(\mu_i(\boldsymbol{x}^{(t+1)}|\mathcal{D}^{1:t}), \sigma^2(\boldsymbol{x}^{(t+1)}|\mathcal{D}^{1:t})) \ , \forall i \in I \ , \tag{8}$$

where

$$\mu_i(\boldsymbol{x}^{(t+1)}|\mathcal{D}^{1:t}) = \boldsymbol{k}_i(\boldsymbol{x}^{(t+1)})^T K_i^{-1} o_i^{1:t} \ ,$$
$$\sigma_i^2(\boldsymbol{x}^{(t+1)}|\mathcal{D}^{1:t}) = k_i(\boldsymbol{x}^{(t+1)}, \boldsymbol{x}^{(t+1)}) - \boldsymbol{k}_i(\boldsymbol{x}^{(t+1)})^T K_i^{-1} \boldsymbol{k}_i(\boldsymbol{x}^{(t+1)}) \ ,$$

with

$$\boldsymbol{k}_i(\boldsymbol{x}^{(t+1)}) = (k_i(\boldsymbol{x}^{(t+1)}, \boldsymbol{x}^{(1)}), \ldots, k_i(\boldsymbol{x}^{(t+1)}, \boldsymbol{x}^{(t)})) \ ,$$
$$k_i(\boldsymbol{x}, \boldsymbol{x}') = v_i \delta(\boldsymbol{x} - \boldsymbol{x}') + c_i \exp\left(- \frac{(\boldsymbol{x} - \boldsymbol{x}')^T D_i (\boldsymbol{x} - \boldsymbol{x}')}{2} \right) .$$

One can observe that the kernel function $k_i(\cdot, \cdot)$ is a combination of the white kernel and the scaled squared exponential kernel to explain the noise-component of the observations. The kernel's hyperparameters v_i, c_i, and the diagonal matrix D_i are tuned by maximizing the GP log-marginal likelihood.

As mentiond in Section 2, $\mathbb{E}[\boldsymbol{o}(\boldsymbol{x})] = \boldsymbol{u}(\boldsymbol{x})$. Thus, according to our GP model, for all $i \in I$, $u_i(\boldsymbol{x}_i, \boldsymbol{x}_{\text{-}i})$ is approximated by

$$u_i(\boldsymbol{x}_i, \boldsymbol{x}_{\text{-}i}) \approx \mathbb{E}[o_i(\boldsymbol{x})|\mathcal{D}^{1:t}] = \mu_i(\boldsymbol{x}|\mathcal{D}^{1:t}) \ . \tag{9}$$

Next, we show how GPs can be used to approximate $u_i(\mathcal{B}(\boldsymbol{x}), \boldsymbol{x}_{\text{-}i})$.

Approximating $u_i(\mathcal{B}(\boldsymbol{x}), \boldsymbol{x}_{\text{-}i})$. This value corresponds to the maximum of the set of values

$$U_i(\boldsymbol{x}_{\text{-}i}) \stackrel{\text{def}}{=} \{u_i(\boldsymbol{x}'_i, \boldsymbol{x}_{\text{-}i}) \mid \boldsymbol{x}'_i \in \mathcal{X}_i\} \subset \mathbb{R} \ .$$

Assuming these values are finite (bounded), one may recover its maximum using its mean and standard deviation, as shown in the motivating example below.

Example 1. Let $U_i(\boldsymbol{x}_{\text{-}i})$ be the support of a uniform distribution $\mathcal{U}([a, b])$, then the maximum of this set, which is b and in our setup it is $u_i(\mathcal{B}(\boldsymbol{x}), \boldsymbol{x}_{\text{-}i})$, can be recovered from the distribution mean $(a + b)/2$ and standard deviation $(b - a)/\sqrt{12}$ as

$$\frac{a+b}{2} + \gamma \frac{b-a}{\sqrt{12}}$$

where $\gamma = \sqrt{3}$.

Similarly, we approximate $u_i(\mathcal{B}(\boldsymbol{x}), \boldsymbol{x}_{\text{-}i})$, the payoff of i's best response to $\boldsymbol{x}_{\text{-}i}$, according to our GP models as

$$u_i(\mathcal{B}(\boldsymbol{x}), \boldsymbol{x}_{\text{-}i}) \approx \bar{\mu}_i(\boldsymbol{x}|\mathcal{D}^{1:t}) + \gamma \bar{\sigma}_i(\boldsymbol{x}|\mathcal{D}^{1:t}) \ ,$$

where $\bar{\mu}_i(\boldsymbol{x}|\mathcal{D}^{1:t})$ and $\bar{\sigma}_i(\boldsymbol{x}|\mathcal{D}^{1:t})$ are the mean and standard deviation of $U_i(\boldsymbol{x}_{\text{-}i})$, respectively. Formally, we have

$$\bar{\mu}_i(\boldsymbol{x}|\mathcal{D}^{1:t}) = \mathbb{E}_{\boldsymbol{x}'_i}[\mu_i(\boldsymbol{x}'_i, \boldsymbol{x}_{\text{-}i}|\mathcal{D}^{1:t})] \ , \tag{10}$$
$$\bar{\sigma}_i^2(\boldsymbol{x}|\mathcal{D}^{1:t}) = \mathbb{E}_{\boldsymbol{x}'_i}[(\mu_i(\boldsymbol{x}'_i, \boldsymbol{x}_{\text{-}i}|\mathcal{D}^{1:t}) - \bar{\mu}_i(\boldsymbol{x}|\mathcal{D}^{1:t}))^2] \ , \tag{11}$$

and $\gamma \geq 0$ is a hyperparameter of our algorithm. An appropriate value can be the 99^{th} percentile of the standard normal distribution ≈ 2.33. Provided that the covariance functions $\{k_i\}$ of our GP models are separable, the values of Eq. 10 and Eq. 11 can be computed exactly (in a closed-form), as shown in Sections **??** and **??** of the supplement materials.

Approximating the regret (Eq. 3). Based on the above, in particular Eq. 9, Eq. 10, and Eq. 11, the game-theoretic regret $\epsilon(\boldsymbol{x})$ can be estimated as

$$\hat{\epsilon}(\boldsymbol{x}|\mathcal{D}^{1:t}) = \max_i \bar{\mu}_i(\boldsymbol{x}|\mathcal{D}^{1:t}) + \gamma\bar{\sigma}_i(\boldsymbol{x}|\mathcal{D}^{1:t}) - \mu_i(\boldsymbol{x}|\mathcal{D}^{1:t}) \ . \tag{12}$$

This constitutes the *acquisition function* of our BO framework. In our implementation, we rescale this term by $\bar{\sigma}_i(\boldsymbol{x}|\mathcal{D}^{1:t})$ to have similar sensitivity across $\boldsymbol{x} \in \mathcal{X}$. This improves the performance of the global optimization procedure in its search for the next play $\boldsymbol{x}^{(t+1)}$.

Exploration-Exploitation Trade-off. In general, the acquisition function in the BO framework balances between exploration (sampling from areas in $\mathcal{X}$ of high uncertainty according to the GP model) and exploitation (sampling from potentially optimal areas in $\mathcal{X}$ according to the GP model). One could observe that the acquisition function represented by the estimated regret (Eq. 12) is more exploitative than exploratory. This can be addressed with several policies from the literature [2]. We employ an ε-greedy policy with $\varepsilon \in (0, 1)$,[1] where the next strategic play $\boldsymbol{x}^{(t+1)}$ is chosen with probability $1 - \varepsilon$ according to the approximated regret $\hat{\epsilon}(\cdot|\mathcal{D}^{1:t})$. Otherwise, it is chosen according to the posterior uncertainty $\sigma_i(\cdot|\mathcal{D}^{1:t})$.

Computational Tractability. With the use of separable kernels, we could compute $\bar{\mu}_i(\boldsymbol{x}|\mathcal{D}^{1:t})$ and $\bar{\sigma}_i(\boldsymbol{x}|\mathcal{D}^{1:t})$ exactly and evaluate the approximated regret $\epsilon(\boldsymbol{x}|\hat{\mathcal{D}}^{1:t})$. However, the computational complexity, besides the cost of computing and inverting the covariance matrix K_i, of computing the regret grows quadratically in the number of observations $|\mathcal{D}^{1:t}|$. This could be manageable in an expensive setup where the number observations is well below 100, i.e., $|\mathcal{D}^{1:t}| \leq 100$. Beyond that, one could estimate these values through sampling. Mathematically, for all $i \in I$,

$$\hat{\bar{\mu}}_i(\boldsymbol{x}|\mathcal{D}^{1:t}) = \frac{1}{S}\sum_{s=1}^{S} \mu_i(\hat{\boldsymbol{x}}_i^{(s)}, \boldsymbol{x}_{-i}|\mathcal{D}^{1:t}) \ , \tag{13}$$

$$\hat{\bar{\sigma}}_i^2(\boldsymbol{x}|\mathcal{D}^{1:t}) = \frac{1}{S}\sum_{s=1}^{S} (\mu_i(\hat{\boldsymbol{x}}_i^{(s)}, \boldsymbol{x}_{-i}|\mathcal{D}^{1:t}) - \hat{\bar{\mu}}_i(\boldsymbol{x}|\mathcal{D}^{1:t}))^2 \ , \tag{14}$$

where S is the number of samples, and $\hat{\boldsymbol{x}}_i^{(s)} \sim \mathcal{U}(\mathcal{X}_i)$ or any relevant sampling technique. This approximation can also be used for GPs with inseparable kernels. Note that there exist other methods to approximate the integrals associated

[1] This ε is different from the $\boldsymbol{\varepsilon}$ in $\boldsymbol{\varepsilon}$-NE. The two quantities happen to have the same symbols. We sought to differentiate them with a bold version in the case of $\boldsymbol{\varepsilon}$-NE.

with the inseparable kernel [4]. On the other hand, the noise and randomness introduced by sampling may help towards a better exploration-exploitation trade-off. Our implementation supports both exact and approximate computation of $\bar{\mu}_i(\boldsymbol{x}|\mathcal{D}^{1:t})$ and $\bar{\sigma}_i(\boldsymbol{x}|\mathcal{D}^{1:t})$ and we refer to these algorithmic variants by `BN-exact` and `BN-approx`, respectively. With this at hand, our general-purpose framework for computing Nash equilibria in expensive, black-box, continuous games is now complete. Algorithm 1 summarizes the procedure and we provide an illustration of the same (the exact variant `BN-exact`) in Fig. 2. The figure shows the progress of `BN` variants from 5 to 20 observations. The top subfigures (a and b) show the sampled profiles and their kernel density estimate (KDE) over the action space $\mathcal{X}$. With more iterations, the sampled profiles get closer to the actual NE. The middle subfigures (c and d) show the estimated payoff and approximated regret for player 1 after 5 and 20 observations, respectively. Likewise, the bottom subfgures (e and f) show the same for player 2. Since subfigures (c), (d), (e), and (f) are similar. We describe them collectively. The center heatmap represents the GP's current estimate of player i's payoff $\mu_i(x_1, x_2|\mathcal{D}^{1:t})$ (Eq. 9). The supplots to the left and bottom show the expectation (along with a 95%-confidence interval) of $\mu_i(x_1, x_2|\mathcal{D}^{1:t})$ over x_1 and x_2, respectively. For player 1, the *left* supplot of subfigures (c) and (d) represent $\bar{\mu}_1(\boldsymbol{x}|\mathcal{D}^{1:5})$ (Eq. 10) and $\bar{\mu}_1(\boldsymbol{x}|\mathcal{D}^{1:20})$ in Fig. 2. For player 2, the *bottom* subplots in subfigures (e) and (f) correspond to $\bar{\mu}_2(\boldsymbol{x}|\mathcal{D}^{1:5})$ and $\bar{\mu}_2(\boldsymbol{x}|\mathcal{D}^{1:20})$ in Fig. 2. For player 1 (i.e., subfigures (c) and (d)), this is computed based on the center and the left subplots. For player 2 (i.e., subfigures (e) and (f)), this is computed based on the center and the *right* subplots. One can observe the smoothness of the plots in the figures of `BN-exact` (Fig. 2). For further illustrations, we refer the reader to Section **??** of the supplement materials.

Algorithm 1 Bayesian optimization for Nash Equilibrium (`BN`)

Require:

$\mathcal{D}^{1:t_0}$: initial design (e.g., Latin hypercube design [18])

T: number of iterations

$\varepsilon \in (0, 1)$: probability of exploration

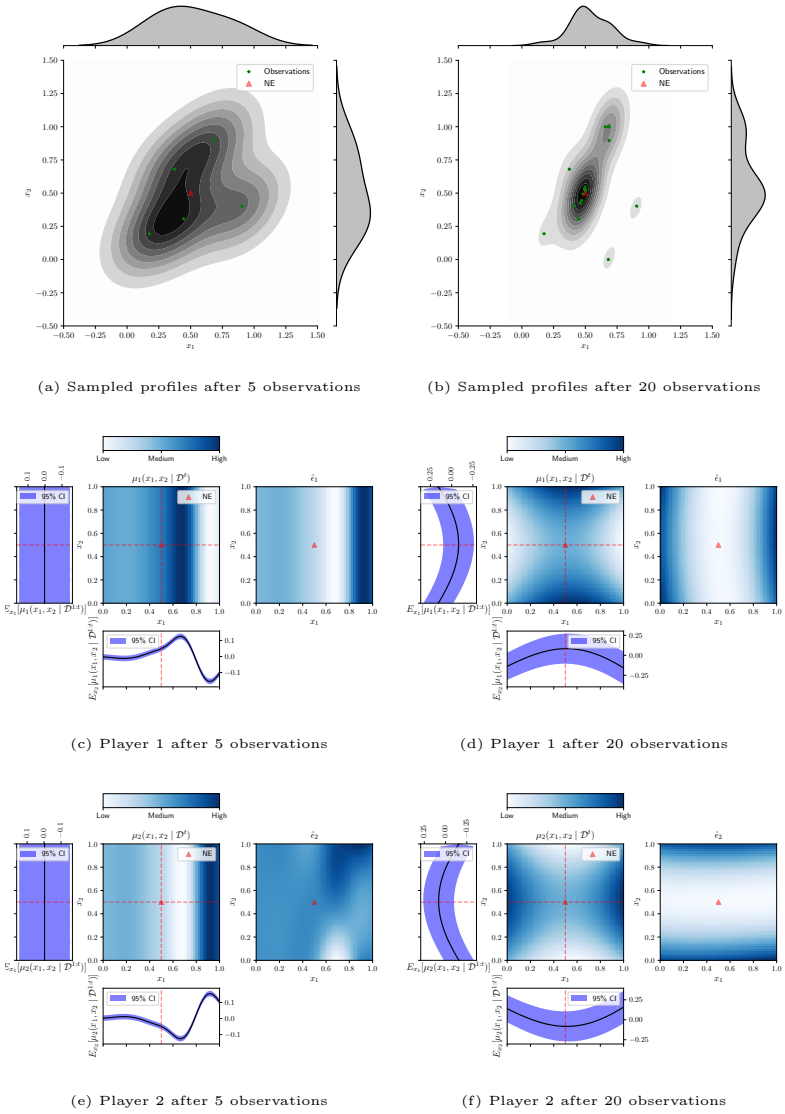
1: **for** $t = 1$ **to** T **do**

2: $\boldsymbol{x}^{(t+1)} \leftarrow \begin{cases} \arg\min_{\boldsymbol{x}\in\mathcal{X}} \hat{\epsilon}(\boldsymbol{x}|\mathcal{D}^{1:t}) , & \text{with probability } 1-\varepsilon; \\ \arg\max_{\boldsymbol{x}\in\mathcal{X}} \max_{i\in I} \sigma_i(\boldsymbol{x}|\mathcal{D}^{1:t}) , & \text{otherwise.} \end{cases}$

3: $\mathcal{D}^{1:t+1} \leftarrow \mathcal{D}^{1:t} \cup \{(\boldsymbol{x}^{(t+1)}, \boldsymbol{o}(\boldsymbol{x}^{(t+1)})\}$

4: **end for**

While the BO framework provides an efficient sampling of the profile space $\mathcal{X}$, one should note that deciding which point (from $\mathcal{D}^{1:T}$) corresponds to the approximate NE, $\boldsymbol{x}(T)$, can be prone to over-fitting [5]. This is not a problem in the standard BO where the best solution $\boldsymbol{x}(T)$ can be decided directly by comparing the values returned by the oracle. This is beyond the scope of this paper and we leave it for future work.



(a) Sampled profiles after 5 observations

(b) Sampled profiles after 20 observations

(c) Player 1 after 5 observations

(d) Player 1 after 20 observations

(e) Player 2 after 5 observations

(f) Player 2 after 20 observations

Fig. 2. Illustration of `BN-exact` after 5 and 20 observations on the zero-sum hyperbolic paraboloid game with NE being $(0.5, 0.5)$. That is, $u_1(x_1, x_2) = (x_2 - 0.5)^2 - (x_1 - 0.5)^2$, and $u_2(x_1, x_2) = -u_1(x_1, x_2)$. $\hat{\epsilon}_i$ denotes $(\bar{\mu}_i(\boldsymbol{x}|\mathcal{D}^{1:t}) + \gamma\bar{\sigma}_i(\boldsymbol{x}|\mathcal{D}^{1:t}) - \mu_i(\boldsymbol{x}|\mathcal{D}^{1:t}))/\bar{\sigma}_i(\boldsymbol{x}|\mathcal{D}^{1:t})$, where $\gamma = 2.32635$, the 99^{th} percentile of the standard normal distribution.

4 Experiments

In this section, we investigate how `BN` performs and compare it with [23]'s Gaussian process method (`GPG`), which discretizes $\mathcal{X}$, and the iterated best-response

(BR) via stochastic search [29]. We are interested in the impact on performance of using a continuous representation, in contrast to the grid that GPG uses. In addition, we are interested in how each algorithm scales and is robust to a noisy payoff function. Our experimental problems are shown in Table 1, the columns show the action space for each player and the given number of function evaluations (#FEs). SADDLE.$\{1, 2, 3\}$ are variants of the zero-sum hyperbolic paraboloid game where the NE $\boldsymbol{x}^*$ is shifted or the game is scaled to a higher-dimensional action space. That is, $u_1(x_1, x_2) = (x_2 - x_2^*)^2 - (x_1 - x_1^*)^2$, and $u_2(x_1, x_2) = -u_1(x_1, x_2)$. The MOP problem is taken from [23]. In line with [7], all algorithms were terminated after $20n_{\mathcal{X}}$ FEs. For comparison with GPG which evaluates at every discretized grid point, with SADDLE.3 we terminated the algorithms after #FEs equal to grid size (120). For experiments with noise, a Gaussian noise was added to the payoff functions with a standard deviation σ_i' that is roughly 0.1 of its range: For MOP, $\sigma_1' = 7.5$ and $\sigma_2' = 3$. For SADDLE variants, $\sigma_1' = \sigma_2' = 0.025$.

Table 1. Problem setup. The columns show the dimensions for each player and the number of fitness evaluations.

Problem	$(\boldsymbol{\mathcal{X}_1}, \boldsymbol{\mathcal{X}_2})$	**NE**	Function evaluations (#FEs)
SADDLE.1	$([0,1],[0,1])$	$(0.5, 0.5)$	40
SADDLE.2	$([0,1],[0,1])$	$(0.3, 0.3)$	40
SADDLE.3	$([0,1]^2,[0,1]^2)$	$([0.5]^2,[0.5]^2)$	120
MOP	$([0,1],[0,1])$	$(0.08093, 1)$	40

BN setup. The source for BN and the supplement materials are available at `https://github.com/ALFA-group`. The algorithm is implemented in Python and uses the Scikit-learn package [22]. The hyperparameters that are not default in BN were set to: A) The bounds of kernel $\{k_i\}$'s hyperparameters were set as follows. $v_i \in [10^{-5}, 10^5]$, $c_i \in [10^{-3}, 10^3]$, and $D_i^{l,l} \in [10^{-2}, 10^2]$. The hyperparameters were tuned with 2 restarts. B) The size of the initial design $|\mathcal{D}^{1:t}|$ is set to $\lfloor \#\text{FEs}/4 \rfloor$. C) The acquisition function (Line 2 of Algorithm 1) is optimized with CMA-ES [11] and an evaluation budget of 250. D) The ε-greedy policy is set with $\varepsilon = 0.05$. E) For BN-approx, $10n_{\mathcal{X}_i}$ samples from the Latin hypercube were used to compute $\hat{\tilde{\mu}}_i(\boldsymbol{x}|\mathcal{D}^{1:t})$ (Eq. 10) and $\hat{\tilde{\sigma}}_i(\boldsymbol{x}|\mathcal{D}^{1:t})$ (Eq. 11).

GPG setup. We tested two variants of the algorithm: GPG-psim and GPG-sur with "psim" and "sur" as the solver criterion, respectively. Similar to BN, $\lfloor \#\text{FEs}/4 \rfloor$ are used as initial points. We used a 31×31 grid for all the problems, except for SADDLE.3 that has 11×11. The rest of the parameters were set to their default settings.

BR setup. In our BR implementation, we used basinhopping from the SciPy package [12], with L-BFGS-B as the local minimization method. BR is prohibitively

expensive on problems with noisy observations, as it requires multiple samples per profile. Therefore, BR is only tested on noiseless experiments for baseline comparison to the methods tailored for expensive games.

Results. Fig. 3 shows the convergence of the best obtained regret ϵ (Eq. 3) as a function of the number of function evaluations over 25 independent runs. Regret is calculated numerically using CMA-ES. We start with SADDLE.1 where the NE is more appropriately placed for the grid used by GPG. In the noiseless version (Fig. 3-a), we observe the lowest regret for the GPG variants. Both BN variants have higher regret than the GPG variants. Still, the BR method has the highest regret. In the noisy SADDLE.1 (Fig. 3-b), the regret is higher for all the methods. But BN variants show a relatively better performance in comparison to the noiseless version of the problem (Fig. 3-a), when noise is added, the BN-approx has the lowest, then BN-exact, both are lower than GPG-psim and GPG-sur. For noiseless SADDLE.2 (Fig. 3-c) the NE is shifted off the grid that GPG uses. Here, we observe the lowest regret for the BN variants. Both BN variants have lower regret than the GPG variants. The BR method has the highest regret. In the noisy SADDLE.2 (Fig. 3-d) all methods have higher regret. Again, the order of the algorithms final regret when noise is added, the BN-approx has the lowest, then BN-exact, both are lower than GPG-psim and GPG-sur. The effect of discretizing $\mathcal{X}$ shows up clearly in the inferior performance of GPG variants over the same problem but with translated (shifted) NEs, namely SADDLE.1 and SADDLE.2.

For noiseless MOP (Fig. 3-e) we observe the lowest regret for the BN variants. The BN-approx has the lowest average regret in the end. Both BN variants have lower regret than the GPG variants. The BR method has the highest regret. When noise is added (Fig. 3-f), all methods have higher regret. There are some changes in the ranking of the algorithms based on final regret, the BN-approx has the lowest, then BN-exact, is slightly lower than GPG-psim, and the highest regret is reported for GPG-sur.

We scaled the easy (for GPG) variant of SADDLE to 4 dimensions (SADDLE.3) and studied its performance when the number of function evaluations are almost equal to the number of GPG's grid points. As 31×31 function evaluations were computationally prohibitive, we used a grid size of 11×11 for GPG. From Fig. 4, we observe the lowest regret for the BN. The GPG versions are not capable on improving the regret, because they are limited to the grid. However the BN variants, as the number of FEs increases, surpass the minimum regret of GPG. From the results, we see that GPG performs well and rapidly finds solutions with low regret when the grid provides the appropriate bias for the search. The GPG starts to struggle when the NE is shifted or the discretization is too wide. This is the gap in performance that the BN variants address. The continuous representation of BN means that it is not limited to the grid and can continue to improve the regret with more fitness evaluations. A drawback of the BN representation is that it can take longer to reach low regret. This tradeoff is acceptable because it is searching the full space and not constrained to only find estimates that occupy the grid. Another benefit of BN is that the memory usage is not dependent on discretization. Setting GPG's grid size to 93×93 produced a memory allocation

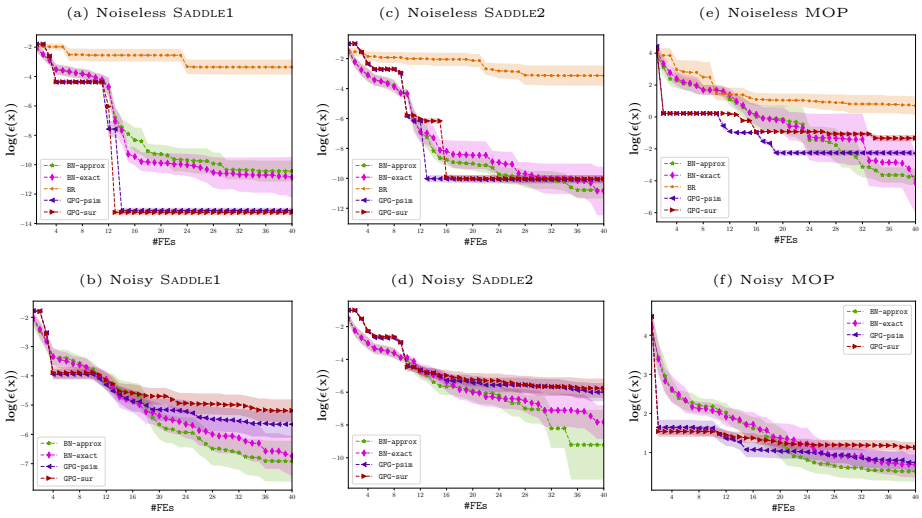


Fig. 3. Regret $\epsilon(\boldsymbol{x})$ convergence on a log scale as a function of the number of function evaluations #`FEs`, obtained using 25 independent runs. The markers indicate the average regret value. The error bands corresponds to one standard deviation.

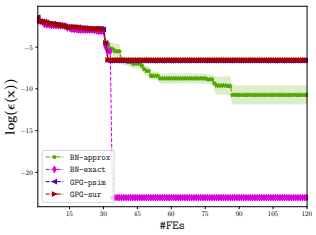


Fig. 4. Regret $\epsilon(\boldsymbol{x})$ convergence on a log scale as a function of the number of function evaluations #`FEs`, obtained using 8 independent runs. The markers indicate the average regret value. The error bands corresponds to one standard deviation.

error on a 24-core Ubuntu machine with a 24-GB RAM. The tradeoff between `BN` and `GPG` is the available #`FEs`, when #`FEs` ≤ 25, `GPG` methods are in general preferable.

5 Conclusion

In this paper, we presented `BN`: a Bayesian framework for computing Nash equilibria for expensive, black-box, continuous games. In contrast to proposed methods in the literature, which either solve a bi-level optimization problem or assume a discretized representation of the strategy space, our framework seeks to jointly learn the full game and estimate the game-theoretic regret on the *full* continuous

strategy space. The approach was validated on a collection of synthetic games and compared to existing methods. We show that `BN` is capable of improving the regret in noisy and high-dimensional games to an extent which hierarchical or discretized methods are not able to reach in an expensive setup. The experiments demonstrated `BN`'s robustness to noise and translated NEs. Secondarily, we observed that the randomness introduced by the empirical computation of the approximated regret (`BN-approx`) can be helpful in directing the search, compared to computing it exactly (`BN-exact`). This is in line with the exploration-exploitation dilemma, in the sense that noisy approximated regret can contribute to the exploration component of our search for NE. Future work will include exploring the parameters of `BN`, incorporating inseparable kernels, and applying it to a simulation-based model for a Cybersecurity application.

References

1. Alberti, R., Nagar, A.K.: Harmony search based algorithm for complete information equilibrium in infinite game. In: Proceedings of the Third International Conference on Soft Computing for Problem Solving. pp. 503–511. Springer (2014)
2. Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multiarmed bandit problem. Machine learning **47**(2-3), 235–256 (2002)
3. Başar, T.: Relaxation techniques and asynchronous algorithms for on-line computation of non-cooperative equilibria. Journal of Economic Dynamics and Control **11**(4), 531–549 (1987)
4. Briol, F.X., Oates, C., Girolami, M., Osborne, M.A.: Frank-wolfe bayesian quadrature: Probabilistic integration with theoretical guarantees. In: Advances in Neural Information Processing Systems. pp. 1162–1170 (2015)
5. Cawley, G.C., Talbot, N.L.: Preventing over-fitting during model selection via bayesian regularisation of the hyper-parameters. Journal of Machine Learning Research **8**(Apr), 841–861 (2007)
6. Debreu, G.: A social equilibrium existence theorem. Proceedings of the National Academy of Sciences **38**(10), 886–893 (1952)
7. Dewancker, I., McCourt, M., Clark, S., Hayes, P., Johnson, A., Ke, G.: A stratified analysis of bayesian optimization methods. arXiv preprint arXiv:1603.09441 (2016)
8. Fearnley, J., Gairing, M., Goldberg, P.W., Savani, R.: Learning equilibria of games via payoff queries. The Journal of Machine Learning Research **16**(1), 1305–1344 (2015)
9. Geem, Z.W.: State-of-the-art in the structure of harmony search algorithm. In: Recent advances in harmony search algorithm, pp. 1–10. Springer (2010)
10. Gibbons, R.: Game theory for applied economists. Princeton University Press (1992)
11. Hansen, N., Müller, S.D., Koumoutsakos, P.: Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). Evolutionary computation **11**(1), 1–18 (2003)
12. Jones, E., Oliphant, T., Peterson, P., et al.: SciPy: Open source scientific tools for Python (2001–), `http://www.scipy.org/`, [Online; accessed <today>]
13. Jordan, P.R., Vorobeychik, Y., Wellman, M.P.: Searching for approximate equilibria in empirical games. In: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 2. pp. 1063–1070. International Foundation for Autonomous Agents and Multiagent Systems (2008)

14. Krawczyk, J.B., Uryasev, S.: Relaxation algorithms to find nash equilibria with economic applications. Environmental Modeling & Assessment **5**(1), 63–73 (2000)
15. Kroer, C., Sandholm, T.: Discretization of continuous action spaces in extensive-form games. In: Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems. pp. 47–56. International Foundation for Autonomous Agents and Multiagent Systems (2015)
16. Li, S., Başar, T.: Distributed algorithms for the computation of noncooperative equilibria. Automatica **23**(4), 523–533 (1987)
17. Lye, K.w., Wing, J.M.: Game strategies in network security. International Journal of Information Security **4**(1-2), 71–86 (2005)
18. McKay, M.D., Beckman, R.J., Conover, W.J.: A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics **42**(1), 55–61 (2000)
19. McKelvey, R.D., McLennan, A.M., Turocy, T.L.: Gambit: Software tools for game theory, version 16.0.1 (2016)
20. Močkus, J.: On bayesian methods for seeking the extremum. In: Optimization Techniques IFIP Technical Conference. pp. 400–404. Springer (1975)
21. Morrow, J.D.: Game theory for political scientists. No. 30: 519.83, Princeton University Press, (1994)
22. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
23. Picheny, V., Binois, M., Habbal, A.: A bayesian optimization approach to find nash equilibria. arXiv preprint arXiv:1611.02440 (2018)
24. Rasmussen, C.E.: Gaussian processes in machine learning. In: Advanced lectures on machine learning, pp. 63–71. Springer (2004)
25. Reeves, D.M.: Generating Trading Agent Strategies: Analytic and Empirical Methods for Infinite and Large Games. Ph.D. thesis, The University of Michigan (2005)
26. Reeves, D.M., Wellman, M.P.: Computing best-response strategies in infinite games of incomplete information. In: Proceedings of the 20th conference on Uncertainty in artificial intelligence. pp. 470–478. AUAI Press (2004)
27. Uryas' ev, S., Rubinstein, R.Y.: On relaxation algorithms in computation of noncooperative equilibria. IEEE Transactions on Automatic Control **39**(6), 1263–1267 (1994)
28. Vorobeychik, Y.: Probabilistic analysis of simulation-based games. ACM Transactions on Modeling and Computer Simulation (TOMACS) **20**(3), 16 (2010)
29. Vorobeychik, Y., Wellman, M.P.: Stochastic search methods for nash equilibrium approximation in simulation-based games. In: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 2. pp. 1055–1062. International Foundation for Autonomous Agents and Multiagent Systems (2008)
30. Vorobeychik, Y., Wellman, M.P., Singh, S.: Learning payoff functions in infinite games. Machine Learning **67**(1-2), 145–168 (2007)
31. Wellman, M.P.: Methods for empirical game-theoretic analysis. In: Proceedings of the national conference on artificial intelligence. vol. 21, p. 1552. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999 (2006)
32. Wellman, M.P., Estelle, J., Singh, S., Vorobeychik, Y., Kiekintveld, C., Soni, V.: Strategic interactions in a supply chain game. Computational Intelligence **21**(1), 1–26 (2005)